Security Assessment

DODO NFT

Apr 28th, 2021

Summary

This report has been prepared for DODO NFT smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Additionally, this audit is based on a premise that all external smart contracts are implemented safely.

The security assessment resulted in 16 informational findings. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	DODO NFT
Platform	BSC
Language	Solidity
Codebase	DODOEX/contractV2/tree/feature/nft
Commits	937ed1586936672b0543fd14f50b2f9bbc761c0d

Audit Summary

Delivery Date	Apr 28, 2021
Audit Methodology	Manual Review
Key Components	

Vulnerability Summary

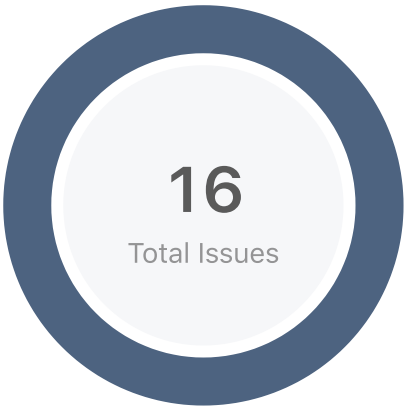
Total Issues	16
Critical	0
Major	0
Minor	0
Informational	16
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
NFT	contracts/CollateralVault/impl/NFTCollateralVault.sol	d3c972f301d955dc69010b29c97e005ededbf149cfea957ca7f64eda83c5e0f0
ICV	contracts/CollateralVault/intf/ICollateralVault.sol	0e568c7a81974199601d4affaee2de28ccde2ffbe75f574667b41febc2b5e319
FDD	contracts/DODOFee/FeeDistributer.sol	7f921ccc4dbc871ee8d5b1dd85396441e50a6111d9cc27ef58a884b11813d441
NFF	contracts/Factory/NFTTokenFactory.sol	b5608abc04d84250a622d581335bdfb243abe09a539ceb11a86da7d84ecff5b7
DOD	contracts/Factory/Registries/DODONFTRegistry.sol	f46317ee431b2ec09411cb16e5f0fd0f0bc5a9cf22dd44ab826d7f42fbe6fa3a
FGF	contracts/GeneralizedFragment/impl/Fragment.sol	c4cddb1d057271074a310ad119558ab778a14fc0bb2c5a5be2a39f4831cad825
IFG	contracts/GeneralizedFragment/intf/IFragment.sol	714138246dac0a5cc3dc18741571575194f02e31a23547906107359dbe8e9176
DOO	contracts/SmartRoute/proxies/DODONFTProxy.sol	d65117d6e741c285633a744cbff9523488125ce2437b2ced0e7328ddfc8bd57c
ERC	contracts/external/ERC1155/ERC1155.sol	faaf3cf25b47955604b0a50581376756fd18affce88309769b36a54d7b96f461
IER	contracts/external/ERC1155/InitializableERC1155.sol	4a3bda390d5a0a10648451641758ed6755bf516b8c8807902210454d9727f2d8

ID	file	SHA256 Checksum
IEC	contracts/external/ERC20/InitializableERC20.sol	20757306d41f8f79c0efba0daca477de3facdaf73c69c51c0d66546232b75ee1

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Minor	0 (0.00%)
Informational	16 (100.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
DOD-01	Zero address	Logical Issue	Informational	Declined
DOD-02	Proper Usage of public and external Type	Gas Optimization	Informational	Resolved
DOD-03	Spell Error	Coding Style	Informational	Resolved
DOD-04	Missing Inheritance	Coding Style	Informational	Resolved
DOD-05	Missing Remove dvm from *REGISTRY*	Logical Issue	Informational	Resolved
DOO-01	Zero address	Logical Issue	Informational	Declined
DOO-02	Missing Check for msg.value	Logical Issue	Informational	Resolved
DOO-03	Function buyout()	Logical Issue	Informational	Resolved
FDD-01	Check Effect Interaction Pattern Violated	Logical Issue	Informational	Resolved
FDD-02	Zero Address	Logical Issue	Informational	Declined
FGF-01	Check Effect Interaction Pattern Violated	Logical Issue	Informational	Resolved
FGF-02	Local Variable Shadowing	Logical Issue	Informational	Resolved
FGF-03	Zero Address	Logical Issue	Informational	Declined
FGF-04	Clear Balance	Logical Issue	Informational	Resolved
NFF-01	Zero address	Logical Issue	Informational	Declined

ID	Title	Category	Severity	Status
NFT-01	Zero Address	Logical Issue	Informational	Partially Resolved

DOD-01 | Zero address

Category	Severity	Location	Status
Logical Issue	Informational	contracts/Factory/Registries/DODONFTRegistry.sol: 59~65	Declined

Description

There is no validation to check whether `vault`, `fragment`, `quoteToken`, `feeDistributor` and `dvm` are zero addresses.

Recommendation

Consider adding some check for them, like bellow:

```
    function addRegistry(
        address vault,
        address fragment,
        address quoteToken,
        address feeDistributor,
        address dvm
    ) external {
        require(isAdminListed[msg.sender], "ACCESS_DENIED");
        require(vault  != address(0), "vault zero address");
        require(fragment  != address(0), "fragment  zero address");
        require(quoteToken  != address(0), "quoteToken  zero address");
        require(feeDistributor  != address(0), "feeDistributor  zero address");
        require(dvm  != address(0), "dvm  zero address");
        _FRAG_FEE_REGISTRY_[fragment] = feeDistributor;
        _DVM_FEE_REGISTRY_[dvm] = feeDistributor;
        _VAULT_FRAG_REGISTRY_[vault] = fragment;
        _REGISTRY_[fragment][quoteToken].push(dvm);
        emit NewRegistry(vault, fragment, feeDistributor, dvm);
    }
```

Alleviation

No alleviation.

DOD-02 | Proper Usage of public and external Type

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/Factory/Registries/DODONFTRegistry.sol: 85, 89	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays `external` functions are more efficient than `public` functions.

Recommendation

Consider using the `external` attribute for these functions.

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

DOD-03 | Spell Error

Category	Severity	Location	Status
Coding Style	Informational	contracts/Factory/Registries/DODONFTRegistry.sol: 85, 89	Resolved

Description

There is a spell error for `addAmindList`, it should be `addAdminList`. By the way, changing from `removeWhiteList` to `removeAdminList` will make sense.

Recommendation

Consider making changes for them, like below:

```
    function addAdminList (address contractAddr) public onlyOwner {
        isAdminListed[contractAddr] = true;
    }

    function removeAdminList (address contractAddr) public onlyOwner {
        isAdminListed[contractAddr] = false;
    }
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

DOD-04 | Missing Inheritance

Category	Severity	Location	Status
Coding Style	Informational	contracts/Factory/Registries/DODONFTRegistry.sol: 29	Resolved

Description

Missing inheritance from the interface `IDODONFTRegistry`.

Recommendation

Consider inheriting from the interface, like below:

```
contract DODONFTRegistry is InitializableOwnable, IDODONFTRegistry {
...
}
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

DOD-05 | Missing Remove dvm from *REGISTRY*

Category	Severity	Location	Status
Logical Issue	Informational	contracts/Factory/Registries/DODONFTRegistry.sol: 74~78	Resolved

Description

It needs to remove `dvm` from `_REGISTRY_` through function `removeRegistry()`, since it added through function `addRegistry()`.

Recommendation

Consider removing `dvm` from `_REGISTRY_` through function `removeRegistry()`.

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

DOO-01 | Zero address

Category	Severity	Location	Status
Logical Issue	Informational	contracts/SmartRoute/proxies/DODONFTProxy.sol: 69~78, 149, 159	Declined

Description

There is no validation to check whether `cloneFactory`, `weth`, `dodoApproveProxy`, `defaultMaintainer`, `vaultTemplate`, `fragTemplate`, `feeTemplate`, `dvmTemplate`, `mtFeeTemplate`, `nftRegistry` and `fragment` are zero addresses.

Recommendation

Consider adding some check for them. For example:

```
        require(cloneFactory  != address(0), "cloneFactory zero address");
        require(weth  != address(0), "weth zero address");
        require(dodoApproveProxy  != address(0), "dodoApproveProxy zero address");
        require(defaultMaintainer  != address(0), "defaultMaintainer zero address");
        require(vaultTemplate  != address(0), "vaultTemplate zero address");
        require(fragTemplate  != address(0), "fragTemplate zero address");
        require(feeTemplate  != address(0), "feeTemplate zero address");
        require(dvmTemplate  != address(0), "dvmTemplate zero address");
        require(mtFeeTemplate  != address(0), "mtFeeTemplate zero address");
        require(nftRegistry  != address(0), "nftRegistry zero address");
        ...
    function buyout(
        address fragment,uint256 quoteAmount,uint8 flag
    ) external payable preventReentrant {
        require(fragment  != address(0), "fragment zero address");
        ...
    }
    function stakeToFeeDistributor(
        address feeDistributor,uint256 stakeAmount,uint8 flag
    ) external payable preventReentrant {
        require(feeDistributor  != address(0), "feeDistributor zero address");
        ...
    }
```

Alleviation

No alleviation.

DOO-02 | Missing Check for msg.value

Category	Severity	Location	Status
Logical Issue	Informational	contracts/SmartRoute/proxies/DODONFTProxy.sol: 148, 158	Resolved

Description

It's better to check if `msg.value` equals `quoteAmount` or `stakeAmount`, when the value of `flag` is `1`.

Recommendation

Consider adding checks for them, like below:

```
    function buyout(
        address fragment,
        uint256 quoteAmount,
        uint8 flag // 0 - ERC20, 1 - quoteInETH
    ) external payable preventReentrant {
       if(flag == 1){
           require(quoteAmount == msg.value);
         } else {
           require(msg.value == 0);
       }
...
    function stakeToFeeDistributor(
        address feeDistributor,
        uint256 stakeAmount,
        uint8 flag // 0 - ERC20, 1 - ETH
    ) external payable preventReentrant {
       if(flag == 1){
           require(stakeAmount == msg.value);
         } else {
           require(msg.value == 0);
         }
...
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

DOO-03 | Function buyout()

Category	Severity	Location	Status
Logical Issue	Informational	contracts/SmartRoute/proxies/DODONFTProxy.sol: 148	Resolved

Description

It is possible that Fragment contains more quote tokens than the amount that all fragments valued, for example, user paid more than the required buyout fee or withdraw more quote tokens from `dvm`.

Recommendation

Consider adding some check or refunding the excess `quoteAmount`.

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

FDD-01 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DODOFee/FeeDistributer.sol: 124~136	Resolved

Description

The order of external call/transfer and storage manipulation must follow check effect interaction pattern.

Recommendation

We advise client to check if storage manipulation is before the external call/transfer operation by considering following modification:

```
    function _claim(address sender, address to) internal {
        uint256 allBase = _USER_BASE_REWARDS_[sender];
        uint256 allQuote = _USER_QUOTE_REWARDS_[sender];
        _BASE_RESERVE_ = _BASE_RESERVE_.sub(allBase);
        _QUOTE_RESERVE_ = _QUOTE_RESERVE_.sub(allQuote);
        _USER_BASE_REWARDS_[sender] = 0;
        _USER_QUOTE_REWARDS_[sender] = 0;
        IERC20(_BASE_TOKEN_).safeTransfer(to, allBase);
        IERC20(_QUOTE_TOKEN_).safeTransfer(to, allQuote);
        emit Claim(sender, allBase, allQuote);
    }
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

FDD-02 | Zero Address

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DODOFee/FeeDistributer.sol: 56~58, 62, 71, 77~81	Declined

Description

There is no validation to check whether `baseToken`, `quoteToken`, `stakeToken` and `to` are zero addresses.

Recommendation

Consider adding some check for them. For example:

```
        require(!_FEE_INITIALIZED_, "ALREADY_INITIALIZED");
        _FEE_INITIALIZED_ = true;
        require(baseToken  != address(0), "baseToken zero address");
        require(quoteToken  != address(0), "quoteToken zero address");
        require(stakeToken  != address(0), "stakeToken zero address");
        _BASE_TOKEN_ = baseToken;
        _QUOTE_TOKEN_ = quoteToken;
        _STAKE_TOKEN_ = stakeToken;
        _STAKE_VAULT_ = address(new StakeVault());
...
    function stake(address to) external {
        require(to  != address(0), "to zero address");
        _updateGlobalState();
        ...
    }
    function claim(address to) external {
        require(to  != address(0), "to zero address");
        _updateGlobalState();
        ...
    }
    function unstake(
        uint256 amount, address to,bool withClaim
    ) external {
        require(to  != address(0), "to zero address");
        require(_SHARES_[msg.sender] >= amount, "STAKE BALANCE ONT ENOUGH");
        ...
    }
```

Alleviation

No alleviation.

FGF-01 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Informational	contracts/GeneralizedFragment/impl/Fragment.sol: 102~108, 119~122	Resolved

Description

The order of external call/transfer and storage manipulation must follow check effect interaction pattern.

Recommendation

We advise client to check if storage manipulation is before the external call/transfer operation by considering following modification:

```
      ...
      uint256 preOwnerQuote = DecimalMath.mulFloor(_BUYOUT_PRICE_,
balances[_VAULT_PRE_OWNER_]);
      _clearBalance(_VAULT_PRE_OWNER_);
      IERC20(_QUOTE_).safeTransfer(_VAULT_PRE_OWNER_, preOwnerQuote);

      uint256 newOwnerQuote = DecimalMath.mulFloor(_BUYOUT_PRICE_,
balances[newVaultOwner]);
      _clearBalance(newVaultOwner);
      IERC20(_QUOTE_).safeTransfer(newVaultOwner, newOwnerQuote);
      ...
      uint256 baseAmount = balances[msg.sender];
      uint256 quoteAmount = DecimalMath.mulFloor(_BUYOUT_PRICE_, baseAmount);
      _clearBalance(msg.sender);
      IERC20(_QUOTE_).safeTransfer(to, quoteAmount);
     ...
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

FGF-02 | Local Variable Shadowing

Category	Severity	Location	Status
Logical Issue	Informational	contracts/GeneralizedFragment/impl/Fragment.sol: 50	Resolved

Description

Local variable `totalSupply` shadows `InitializableERC20.totalSupply`.

Recommendation

Consider renaming the local variable `totalSupply` as `_totalSupply`, like below:

```
function init(
  address dvm,
  address vaultPreOwner,
  address collateralVault,
  uint256 _totalSupply,
  uint256 ownerRatio,
  uint256 buyoutTimestamp
) external {
    ...
    super.init(address(this), _totalSupply, name, symbol, decimals);

    // init FRAG distribution
    uint256 vaultPreOwnerBalance = DecimalMath.mulFloor(_totalSupply, ownerRatio);
    _transfer(address(this), _VAULT_PRE_OWNER_, vaultPreOwnerBalance);
    _transfer(address(this), _DVM_, _totalSupply.sub(vaultPreOwnerBalance));
    ...
}
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

FGF-03 | Zero Address

Category	Severity	Location	Status
Logical Issue	Informational	contracts/GeneralizedFragment/impl/Fragment.sol: 58~61, 81, 116	Declined

Description

There is no validation to check whether `dvm`, `vaultPreOwner`, `collateralVault`, `newVaultOwner` and `to` are zero addresses.

Recommendation

Consider adding some check for them, like below:

```
        ...
        require(!_FRAG_INITIALIZED_, "DODOFragment: ALREADY_INITIALIZED");
        _FRAG_INITIALIZED_ = true;
        require(dvm  != address(0), "dvm zero address");
        require(vaultPreOwner  != address(0), "vaultPreOwner zero address");
        require(collateralVault  != address(0), "collateralVault zero address");
        // init local variables
        _DVM_ = dvm;
        _QUOTE_ = IDVM(_DVM_)._QUOTE_TOKEN_();
        _VAULT_PRE_OWNER_ = vaultPreOwner;
        _COLLATERAL_VAULT_ = collateralVault;
        _BUYOUT_TIMESTAMP_ = buyoutTimestamp;
       ...
    function buyout(address newVaultOwner) external {
        require(_BUYOUT_TIMESTAMP_ != 0, "DODOFragment: NOT_SUPPORT_BUYOUT");
        require(newVaultOwner  != address(0), "newVaultOwner zero address");
...
    function redeem(address to, bytes calldata data) external {
        require(_IS_BUYOUT_, "DODOFragment: NEED_BUYOUT");
        require(to  != address(0), "to zero address");
...
```

Alleviation

No alleviation.

FGF-04 | Clear Balance

Category	Severity	Location	Status
Logical Issue	Informational	contracts/GeneralizedFragment/impl/Fragment.sol: 145	Resolved

Description

`balances[address(0)]` just records the last clear balance, accumulating all clear balance will make sense.

Recommendation

Consider accumulating all the clear balances, like below:

```
...
balances[address(0)] = balances[address(0)].add(clearBalance);
...
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

NFF-01 | Zero address

Category	Severity	Location	Status
Logical Issue	Informational	contracts/Factory/NFTTokenFactory.sol: 43~45	Declined

Description

There is no validation to check whether `cloneFactory`, `erc721Template` and `erc1155Tempalte` are zero addresses.

Recommendation

Consider adding some check for them. For example:

```
...
require(cloneFactory  != address(0), "cloneFactory zero address");
require(erc721Template  != address(0), "erc721Template zero address");
require(erc1155Tempalte  != address(0), "erc1155Tempalte zero address");
_CLONE_FACTORY_ = cloneFactory;
_ERC721_TEMPLATE_ = erc721Template;
_ERC1155_TEMPLATE_ = erc1155Tempalte;
...
```

Alleviation

No alleviation.

NFT-01 | Zero Address

Category	Severity	Location	Status
Logical Issue	Informational	contracts/CollateralVault/impl/NFTCollateralVault.sol: 38, 50, 62, 67	Partially Resolved

Description

There is no validation to check whether `owner`, `newOwner` and `nftContract` are zero addresses.

Recommendation

Consider adding some check, like below:

```
    ...
        require(owner  != address(0), "DODONftVault: owner zero address");
        initOwner(owner);
    ...
    function directTransferOwnership(address newOwner) external onlyOwner {
        require(newOwner != address(0), "DODONftVault: newOwner zero address");
        _OWNER_ = newOwner;
        emit OwnershipTransferred(_OWNER_, newOwner);
    }
    ...
    function withdrawERC721(address nftContract, uint256 tokenId) external onlyOwner {
        require(nftContract != address(0), "DODONftVault: nftContract zero address");
        ...
    }
    ...
    function withdrawERC1155(address nftContract, uint256[] memory tokenIds, uint256[]
memory amounts) external onlyOwner {
        require(nftContract != address(0), "DODONftVault: nftContract zero address");
        ...
    }
```

Alleviation

The development team heeded our advice and resolved this issue in commit `a4f78f0d06c55a64f5faa90c3b49e28294779285`.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.